Why AI Frameworks Need (not only) RDMA?

With Design and Implementation Experience of Networking Support on TensorFlow GDR, Apache MXNet, WeChat Amber, and Tencent Angel

Bairen Yi (byi@connect.ust.hk)
Jingrong Chen, Junxue Zhang, Jiacheng Xia, Li Chen, and Kai Chen
SING Group, WHAT Lab, Big Data Institute
Hong Kong University of Science & Technology

A Perspective from (non-HPC) System and Networking Community

- Prof. Kai Chen and System & Networking Lab @ HKUST
- Research interests include networked systems design and implementation, data center networks, data centric networking, and cloud & big data systems
- 5 papers in NSDI’15-18, 4 papers in SIGCOMM’15-17
- Collaborations with industrial partners including Tencent & Huawei on real-world systems in AI, Big Data, and Cloud

Industrial Experience from an Academic Lab?

- Worked on real-world AI & Big Data systems, like CoDA with Huawei (2015), Tencent Angel (2016), WeChat Amber (2017)

- Contributed network optimisation patches to several open source projects, including TensorFlow & Apache MXNet

- A recently funded startup in Beijing, providing commodity & high performance data center networking solutions to AI teams of data scientists, developers, and operations

Agenda

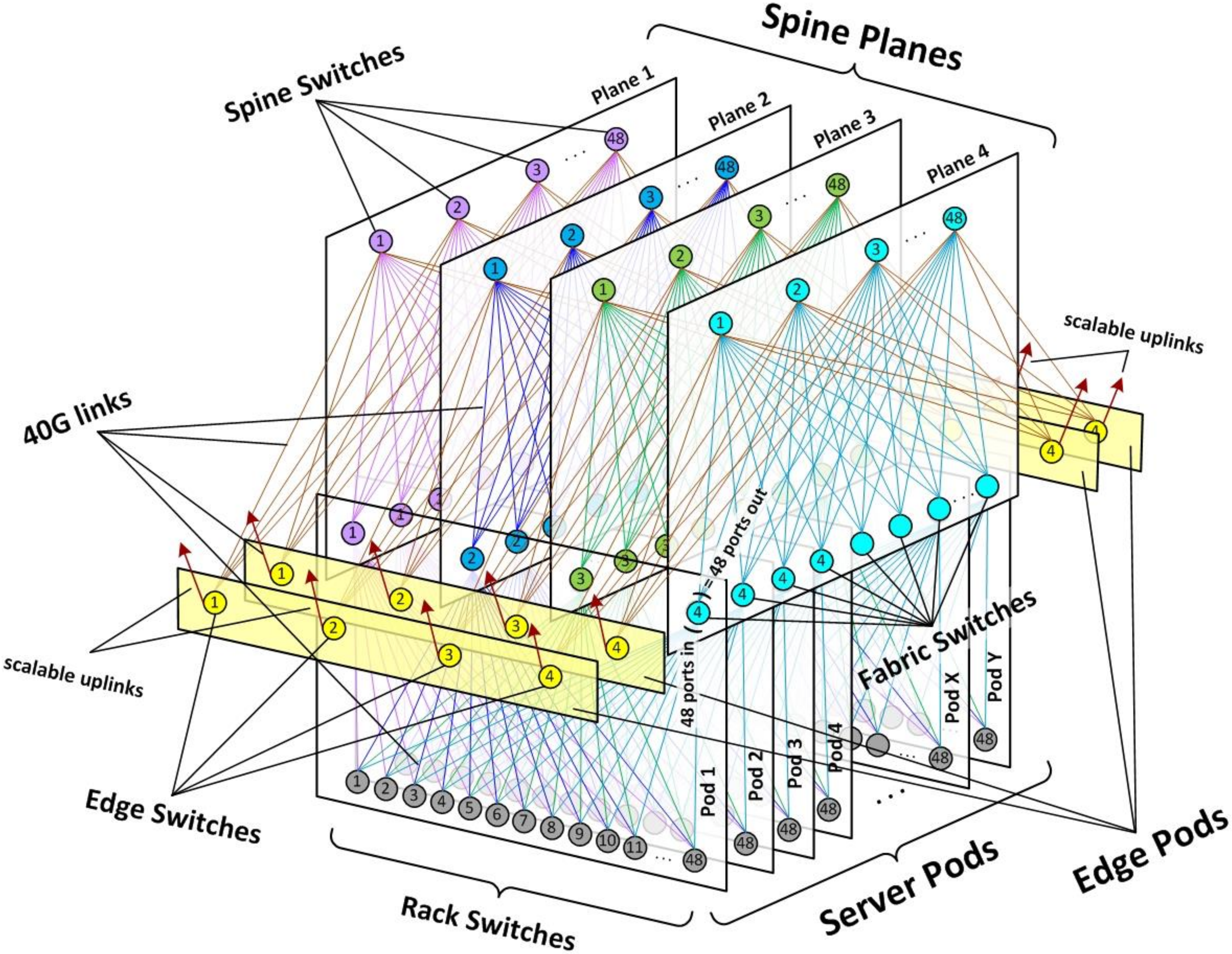
- A glance to commodity data center networking
- Convergence of networking in cloud data centers
- Anti-patterns with high performance networks
- End-to-end design principles for AI frameworks

The Rise of Cloud Computing

40 Azure Regions across the World

Data Centers

What does data center network look like?



Spine Planes
Spine Switches
Plane 1
Plane 2
Plane 3
Plane 4
scalable uplinks
40G links
48 ports in ( ) = 48 ports out
Fabric Switches
scalable uplinks
Pod 1
Pod 2
Pod 3
Pod 4
Pod X
Pod Y
Edge Switches
Rack Switches
Server Pods
Edge Pods

What Modern Data Center Networks Offer

- High throughput: single connection at line rate
- Low latency: 99.9% tail latency within 200 μs*
- Scalability: >100,000 nodes in routable IP network
- Commodity: <$100($500) 25(100) GbE per port

*RDMA over Commodity Ethernet at Scale, SIGCOMM'16

Convergence of Data Center Networking Technologies

- InfiniBand, OmniPath, Fibre Channel, and PLX (PCIe Switch)
- Replacing 4 networks (and switches) with a single Ethernet
- Convergence of networking applications to IP as well: computation, storage, messaging, and remote management
- (Routable) RDMA over Converged Ethernet (RoCEv2)

Evolution of Network I/O

- Latency dropped from ~10 ms to ~10 μs
- From kernel to user space (VMA, DPDK, Onload)
- From software to hardware (RoCE, iWARP)
- Reduced CPU load for better performance

Software Anti-Patterns

- High performance networking costs and we can only afford commodity Ethernet (a.k.a. AWS VPC)
- Network communication hurts performance and we need to avoid communication as much as possible
- Either high-level APIs with poor performance or low-level APIs with high performance; not both

Networking APIs Revisited

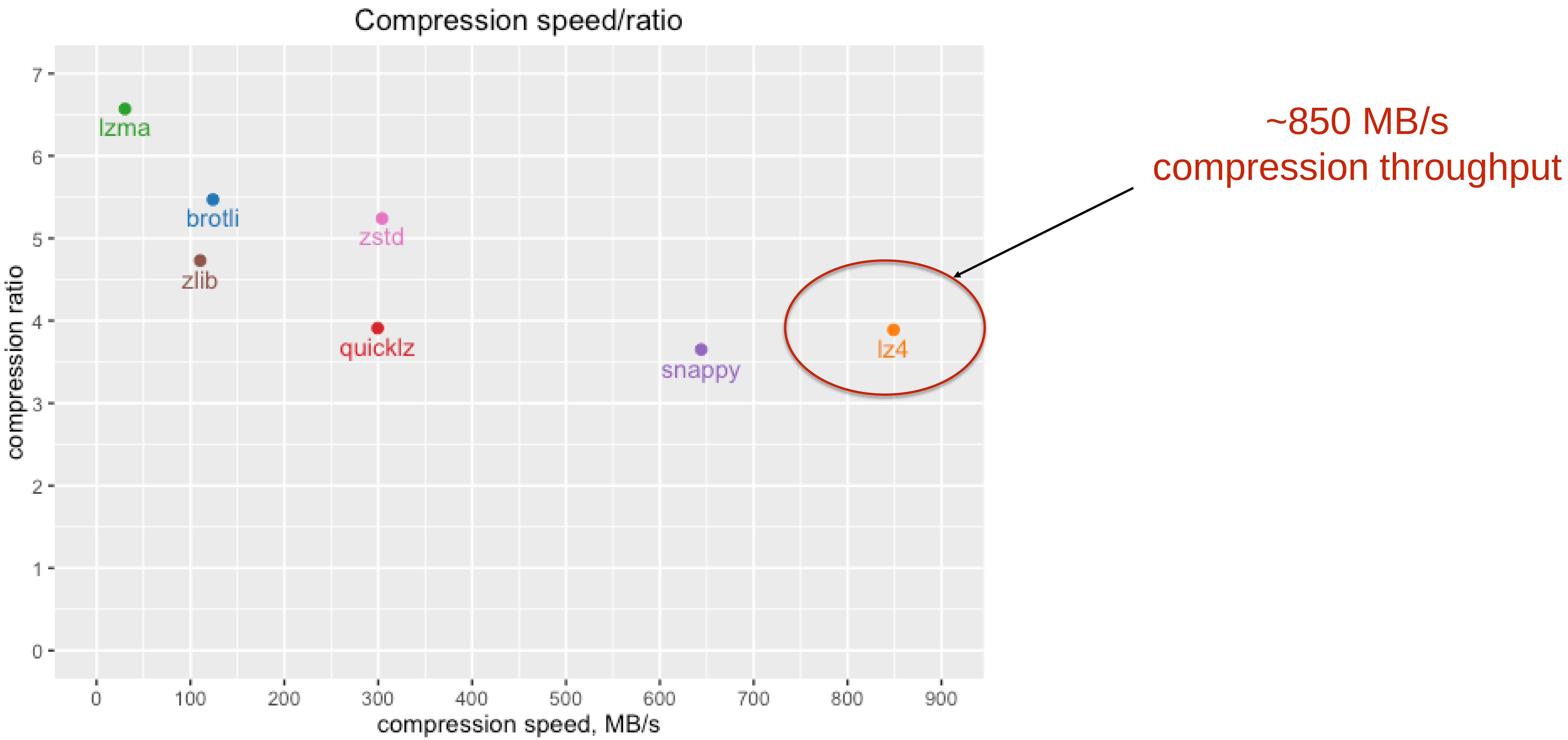
- Messaging libraries: socket, 0MQ, Netty, Akka
- RPC libraries: gRPC, Thrift, Dubbo, brpc
- Encoding libraries: protobuf, thrift, kryo, flatbuffers
- Compression libraries: zlib, snappy, lz4

	FlatBuffers (binary)	Protocol Buffers LITE
Decode + Traverse + Dealloc (1 million times, seconds)	0.08	302
Decode / Traverse / Dealloc (breakdown)	0 / 0.08 / 0	220 / 0.15 / 81
Encode (1 million times, seconds)	3.2	185
Wire format size (normal / zlib, bytes)	344 / 220	228 / 174
Memory needed to store decoded wire (bytes / blocks)	0 / 0	760 / 20
Transient memory allocated during decode (KB)	0	1
Generated source code size (KB)	4	61
Field access in handwritten traversal code	typed accessors	typed accessors
Library source code (KB)	15	some subset of 3800

~100 MB/s
encoding throughput

Lesson Learnt: Do not encode your data when your network >100 Gbps

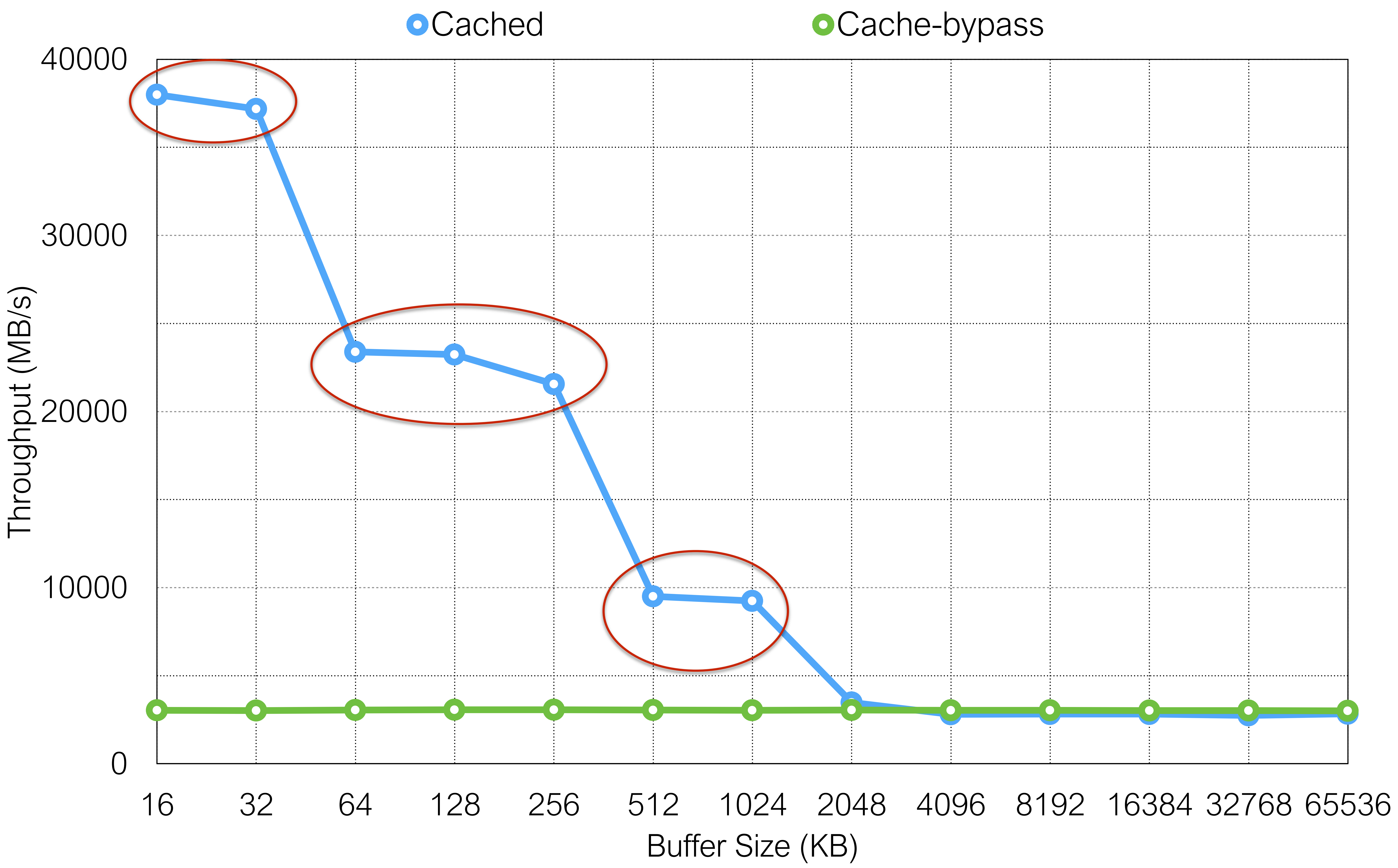
Taken from: https://google.github.io/flatbuffers/flatbuffers_benchmarks.html



Lesson Learnt: Do not compress your data
when your network >100 Gbps

Taken from: https://www.percona.com/blog/2016/04/13/evaluating-database-compression-methods-update/

How about Memory Copy?



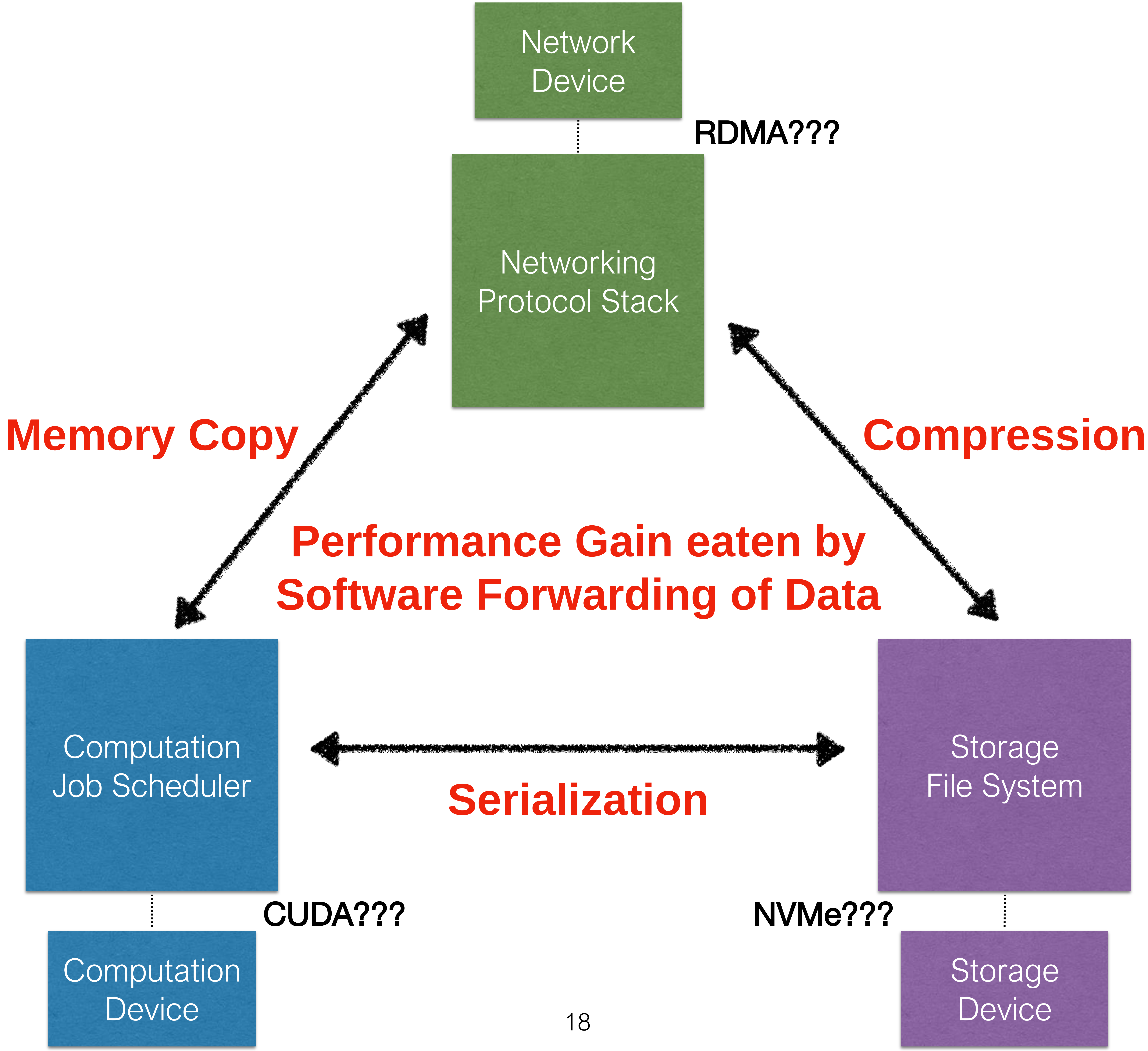
Cached Writes: 37.7 GB/s at 16 KB, 3.0 GB/s >L3 cache
Cache Bypassed Writes: 3.0 GB/s at all buffer sizes

Taken from: http://zsmith.co/bandwidth.html

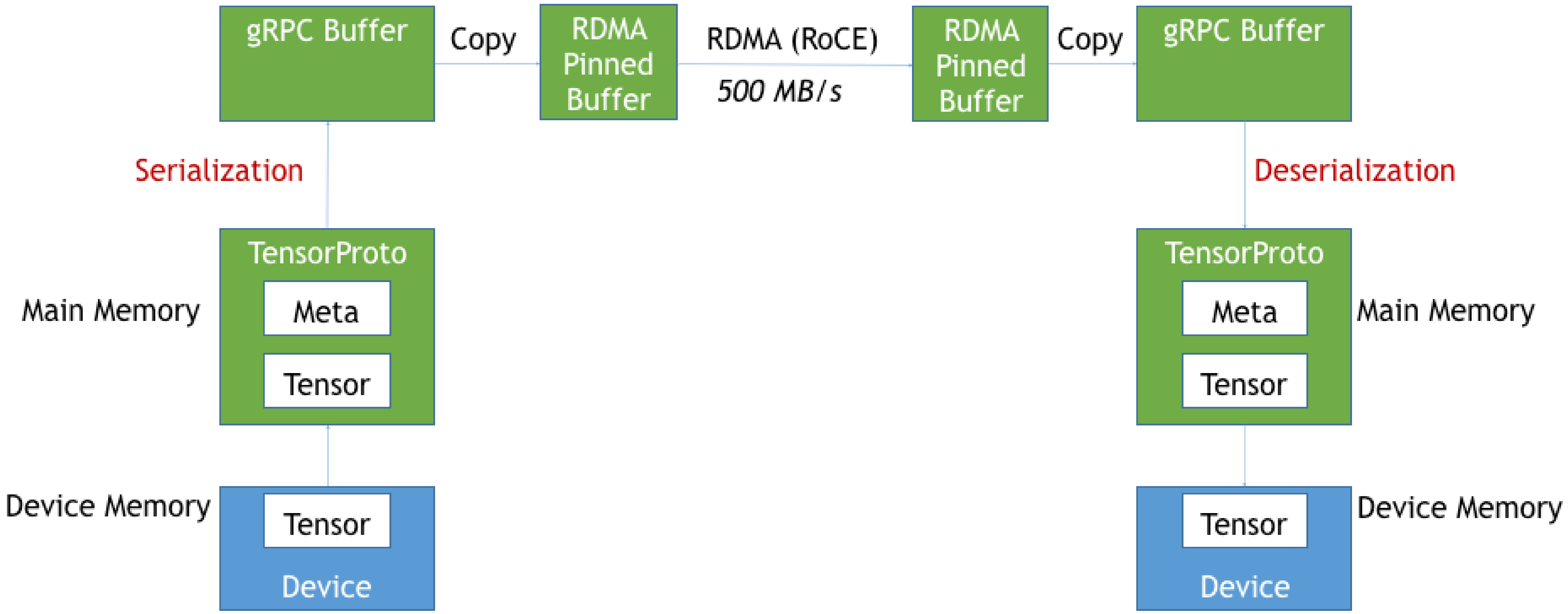
AI Applications are Bottlenecked by its Anti-Patterns

- The performance of Spark 1.4 increases only **2%** by medium even if the network is **infinitely** fast*!
- Encoding, compression, serialisation, and memory copying take the most CPU cycles, not networking (nor disk I/O; about 20% better if it's infinitely fast)
- Software architecture makes CPU its bottleneck

*Making Sense of Performance in Data Analytics Frameworks, NSDI'15


Network Device
RDMA???
Networking Protocol Stack
Memory Copy
Compression
Performance Gain eaten by Software Forwarding of Data
Computation Job Scheduler
Serialization
Storage File System
CUDA???
NVMe???
Computation Device
Storage Device

Yahoo's TensorFlow RDMA

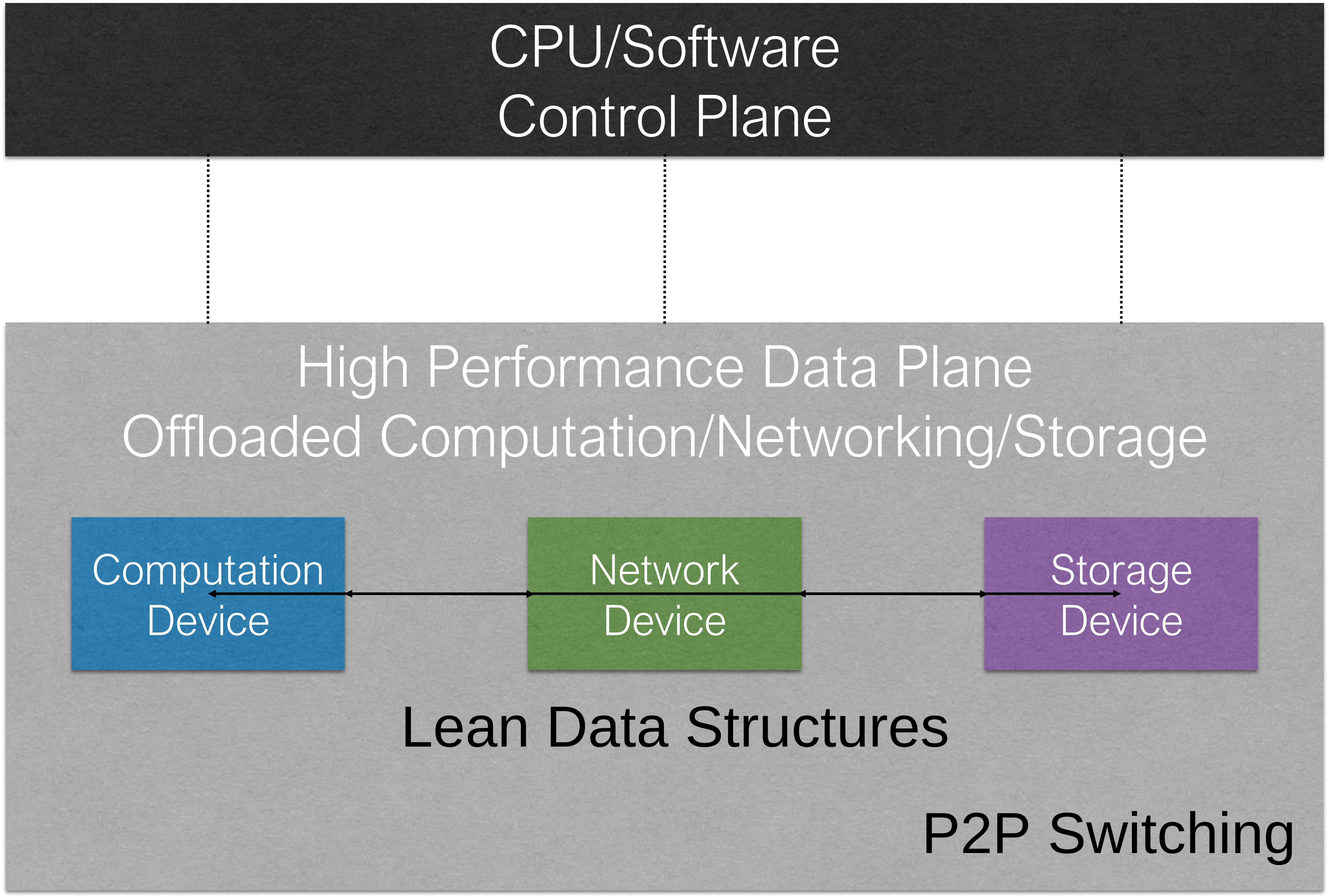


* Removing Ser/Des gives 1.5 GB/s throughput

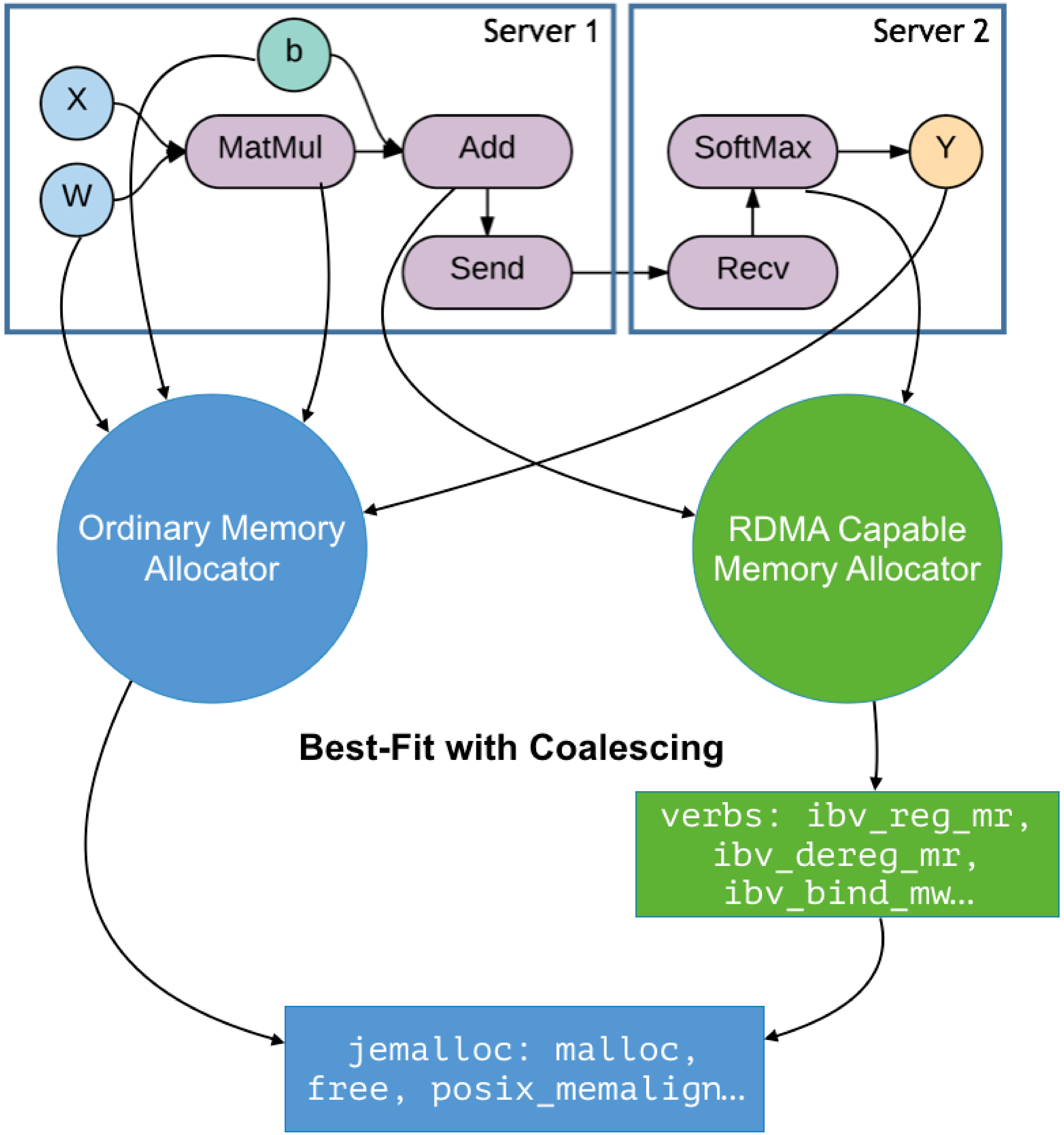
0-Copy Dataflow

- The end-to-end offloaded dataflow pipeline: NVMe storage, RDMA networks, and GPU accelerators
- Lesson learnt from network switches: we need to separate control plane and data plane
- CPU/software for flexible control plane, hardware offloading for high performance data plane

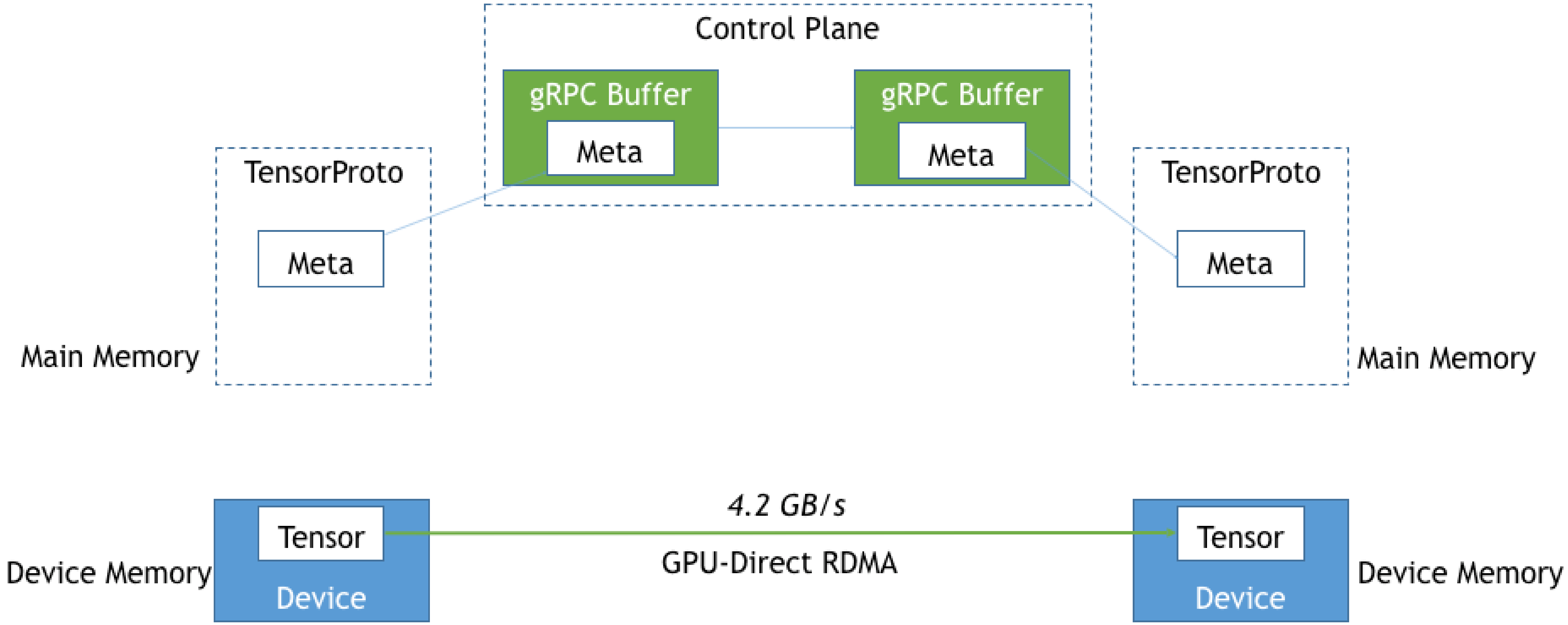
Disaggregated Architecture



Dataflow partition for distributed execution



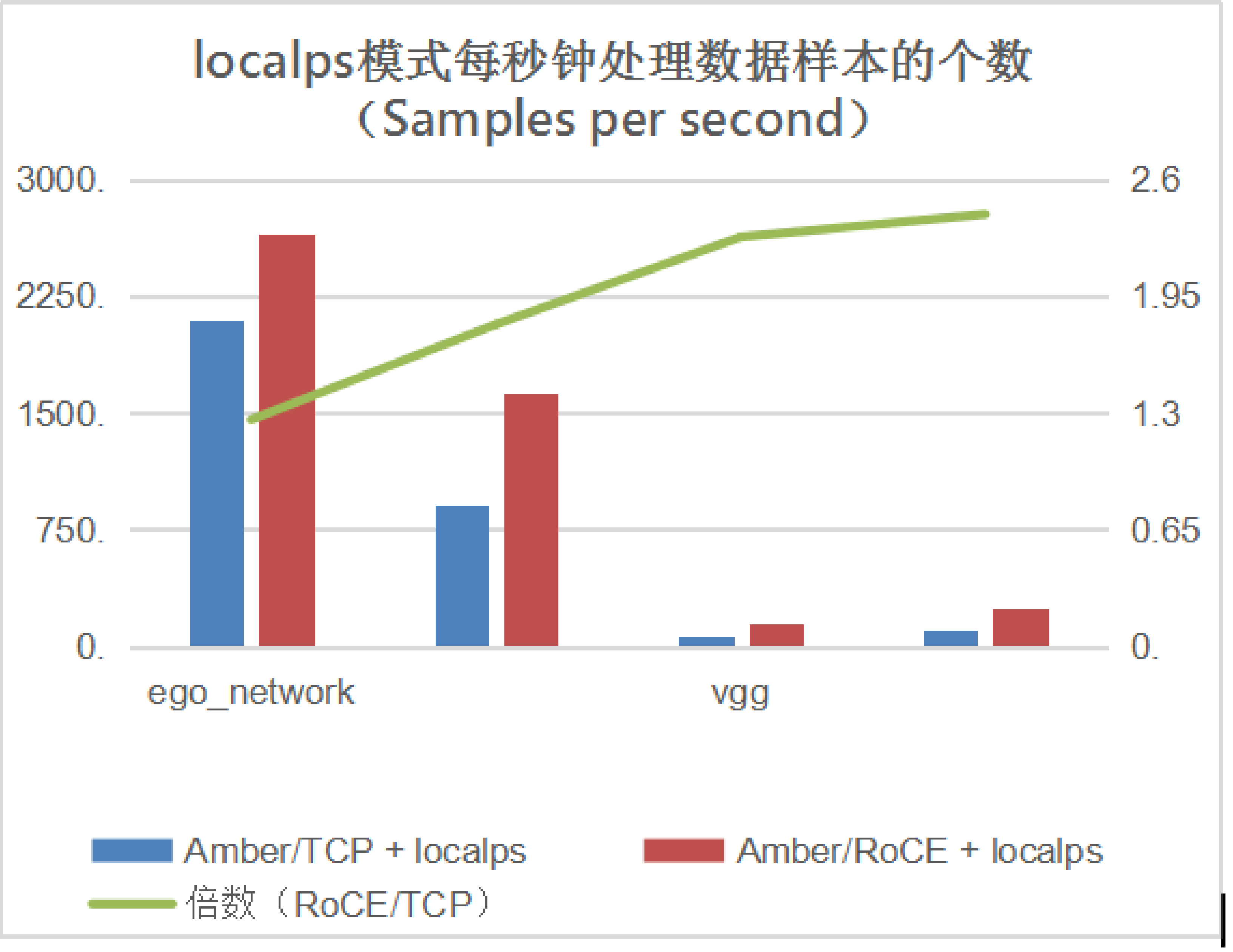
TensorFlow GDR

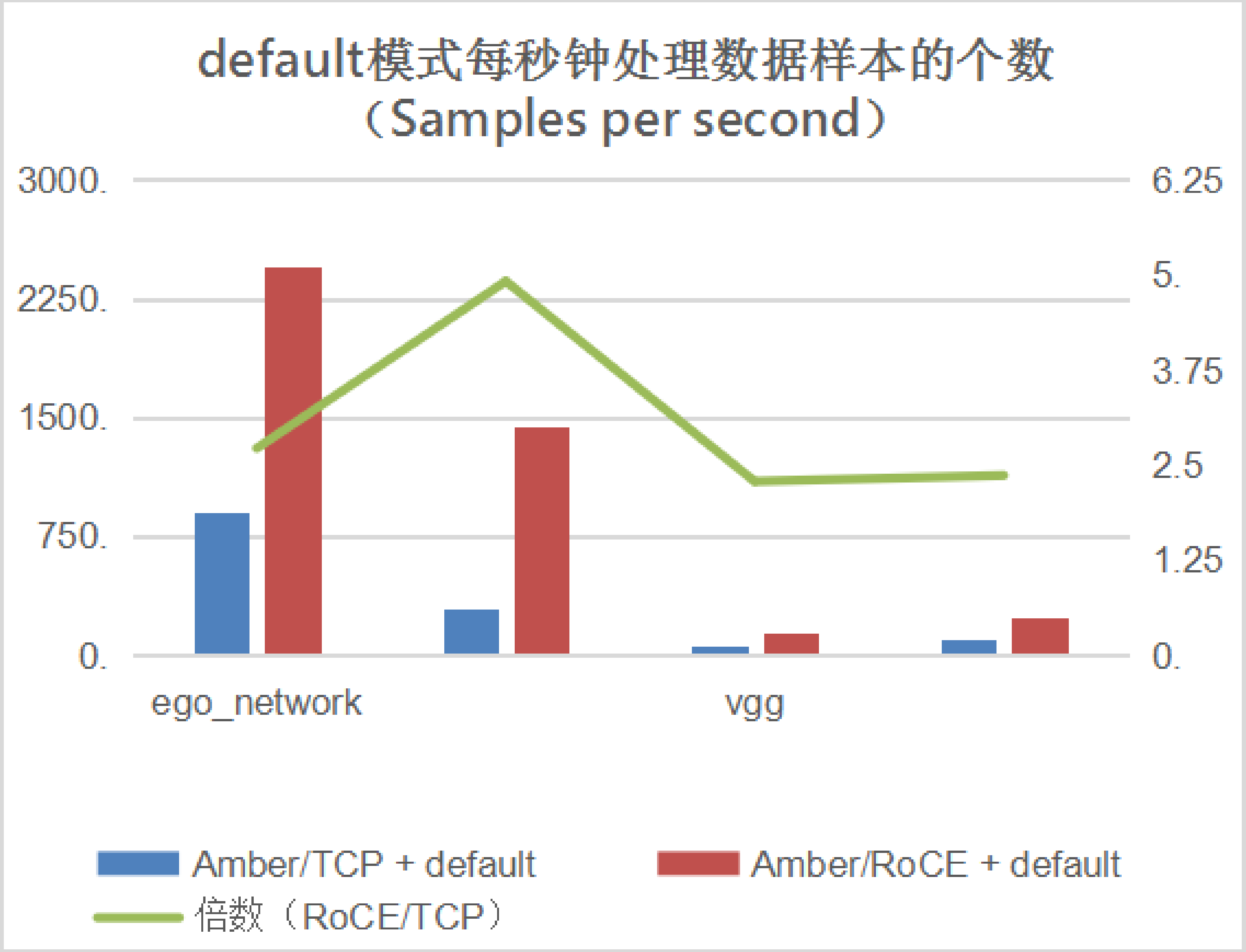


TensorFlow GDR

- We kept 100% gRPC calls without introducing significant overhead compared to pure RDMA
- Easy to fallback to gRPC with mixed deployment
- Less code changes compared to verbs with even more features (<1,000 lines of C++, mostly on GPU Direct RDMA and RDMA memory management)

WeChat's Amber w/ RDMA





Taken from: http://www.infoq.com/cn/news/2017/08/RoCE-wechat-amber

WeChat's Amber w/ RDMA

Scalability Ratio (0MQ for TCP)

	Ego Network	Deep Conversation	Object Recognition
Amber/TCP	0.34	0.41	0.42
Amber/RoCE	0.98	0.99	1.00

Taken from: http://www.infoq.com/cn/news/2017/08/RoCE-wechat-amber

To Copy or Not To Copy

- RDMA messages need to be registered (pinned) through **ibv_reg_mr** before send/recv
- Pinning memory pages through **get_user_pages** in kernel is costly, e.s.p. frequently for small buffers
- Typically we introduce transmitting/receiving side ring buffers w/ huge pages for RDMA buffer reuse
- Buffer bloat and extra latency introduced in copy

Looking Forward

- Unified Virtual Memory (UVM) in CUDA 6
- On Demand Paging (ODP) in OFED 4
- MSG_ZEROCOPY by Google in Linux 4.14
- Heterogeneous Memory Management (HMM) for universal coherent host+device memory space

Conclusion

- Embrace end-to-end principle designing AI frameworks for data intensive applications

- Revisit old operating system concepts and learn how to write low level programs for your hardware

- Combining high-level APIs with efficient offloading actually works (as long as hardware does all the heavy lifting and software only in the control plane)

Questions?